GEORGIA INSTITUTE OF TECHNOLOGY

School of Electrical and Computer Engineering

ECE-6276-A DSP Hardware System Design (Fall 2018)

Lab 4: Area-optimized Multiplier and Handshake Protocol Design on Datapath

Assigned Date	**: Friday, September 21, 2018**
Due Date	**: 23:55, Thursday, September 27, 2018**

1 Introduction

In this lab, we will be optimizing the area of the `multiplier_updated.vhd` in Lab 3. As you can see in the utilization report of this updated multiplier, the number of used LUTs and Registers as Flip-Flops doesn't reduce much compared to the very first design `multiplier.vhd`. Based on Figure 3 in Lab 3, think of an area-optimized way by shrinking the bitwidth of 3 operands A, B and D, as well as M. Your design should do similar operations as Figure 4 in Lab 3 (Keep same number of pipeline stages).

Also, we will implement a simple handshake protocol on datapath. Think of a producer-consumer handshake protocol in Figure 1. If ready is high, Consumer tells Producer that Consumer is ready to accept new data. On the other side, the Producer generates the next valid data independent of the ready signal. If valid and ready is high at the same cycle, it means the consumer will accept the data at next clock cycle; and the producer is able to generate next new data. However, if the ready signal is low at the cycle where valid is high, the producer must hold the current valid data until the consumer accepts it.

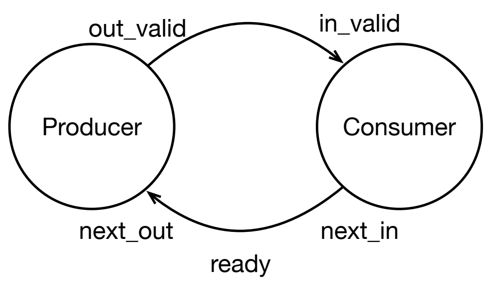

Figure 1 Producer-Consumer Handshake Protocol

The reason why valid is important is that different architectures will lead to different latencies. Thus, the outside world should know when the output data is to be captured. So, let's say Engineer A has designed his system so that latency is 8 cycles. Engineer B's design has a latency of 10 cycles. To the outside world (in this case the testbench), this does not matter as the testbench is agnostic to the internal details of the design. It must check only the valid signal.

The reason why ready is important is that different implementations will lead to different capacity of data accepted from the input. Thus, the previous block (producer) should know when it can send the next valid data to the next block (consumer). In the previous design without the ready signal, if the consumer is not able to accept data while the producer keeps generating a series of valid data, the consumer will lose these data packets. If the consumer is not ready, the consumer will put a back-pressure to the producer asking the producer to hold the current data. In a more robust design, each data channel should be accompanied with a valid and ready signal. In this lab, the testbench will receive back pressure from the input side of the design, and the design will also receive back pressure from the output side.

For any stage in the pipeline, the ready (or stall) signal is generated as follows (See Figure 2):

- The stall signal is the inverse of ready signal.
- If next stage is stalled and the current stage has valid data, then the current stage is also stalled.

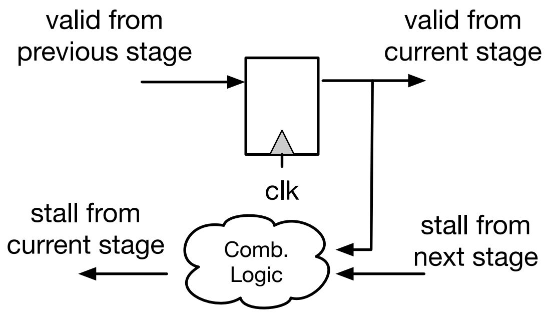

Figure 2 Generation of stall signal

In this lab, we will add valid and ready signals on both the input and output ports of the multiplier. The `next_in` is the ready signal generated by our design. When this `next_in` is asserted (set to logic high), it tells the previous component that the multiplier is ready to accept new data. The `next_out` is the ready signal generated by the next component to our design. When `next_out` is logic low, our multiplier has to hold the valid output until next component accepts it.

2 Instructions

1. Draw the diagram of your updated "two products in a packed word" similar to Figure 3 in Lab 3.

2. Update your design `multiplier_updated.vhd` in Lab 3 based on the diagram above.
3. Use the testbench in Lab 3 to verify the correctness of your design.
4. Move your design into the file `multiplier_lab4.vhd` (Template is given) and add hand-shake protocol.
5. Run the simulation for `multiplier_lab4.vhd` using the provided testbench `tb_multiplier_lab4.vhd` to match your output file `output.txt` with the reference `output_ref.txt`.
6. Run the synthesis and implementation of your design `multiplier_lab4.vhd` in Vivado using the provided constraint file `constraints.xdc`.

3 Deliverables

1. Create a PDF report containing:
   - (20% of grade) Diagram of updated "two products in a packed word".
   - Tables of
     (a) Resources (Only "Slice Logic", "DSP", "IO and GT Specific", "Primitives") from `multiplier_lab4_utilization_placed.rpt`
     (b) Power (Only "Dynamic" and "Static") from `multiplier_lab4_power_routed.rpt`
     (c) Worst Negative Slack (Only "Design Timing Summary") from `multiplier_lab4_timing_summary_routed.rpt`
   - (10% of grade) Answer the question: Why don't we immediately stall the current stage if the next stage is stalled but instead take into consideration of the valid signal? Use scenarios to explain your thought. (Answer should not exceed 6 sentences)

   The name of the PDF should be of the form `lab4_firstname_lastname.pdf` , e.g. `lab4_george_burdell.pdf` .
2. (60% of grade) The completed design file `multiplier_lab4.vhd` . (No latch generated; synchronous reset, sensitivity signal list)
3. The simulated output file `output.txt` in the "run" folder.
4. (10% of grade) The implemented area (utilization), power and timing reports, namely,

   `multiplier_lab4_utilization_placed.rpt`

   `multiplier_lab4_power_routed.rpt`

Move all of these files into a folder called `lab4_firstname_lastname_gtID` . Zip the folder and upload the archive `lab4_firstname_lastname_gtID.zip` on T-Square (eg. `lab4_george_burdell_123456789.zip`). The first name and last name should be all in lower case. **Please strictly follow this naming convention. Otherwise my script will not work and you might get points deduction.**

Note: Late submissions are not accepted. In case of extraordinary circumstances, written permission must be obtained from Dr. Madisetti.